Kubernetes Services for Egress Traffic

6 minute read ✓ page test

Kubernetes `ExternalName` services and Kubernetes services with `Endpoints` let you create a local DNS *alias* to an external service. This DNS alias has the same form as the DNS entries for local services, namely `<service name>.<namespace`

`name>.svc.cluster.local`. DNS aliases provide *location transparency* for your workloads: the workloads can call local and external services in the same way. If at some point in time you decide to deploy the external service inside your cluster, you can just update its Kubernetes service to reference the local version. The workloads will continue to operate without any change.

This task shows that these Kubernetes mechanisms for accessing external services continue to work with Istio. The only configuration step you must perform is to use a TLS mode other than Istio's mutual TLS. The external services are not part of an Istio service mesh so they cannot perform the mutual TLS of Istio. You must set the TLS mode according to the TLS requirements of the external service and according to the way your workload accesses the external service. If your workload issues plain HTTP requests and the external service requires TLS, you may want to perform TLS origination by Istio. If your workload already uses TLS, the traffic is already encrypted and you can just disable Istio’s mutual TLS.

This page describes how Istio can integrate with existing Kubernetes configurations. For new deployments, we recommend following Accessing Egress Services.

While the examples in this task use HTTP protocols, Kubernetes Services for egress traffic work with other protocols as well.

Before you begin

- Setup Istio by following the instructions in the

Installation guide.

> The egress gateway and access logging will be enabled if you install the `demo` configuration profile.

- Deploy the sleep sample app to use as a test source for sending requests. If you have automatic sidecar injection enabled, run the following command to deploy the sample app:

  ```
  $ kubectl apply -f @samples/sleep/sleep.yaml@
  ```

Otherwise, manually inject the sidecar before deploying the `sleep` application with the following command:

```
$ kubectl apply -f <(istioctl kube-inject -f @samples/sleep
/sleep.yaml@)
```

> You can use any pod with `curl` installed as a test source.

- Set the `SOURCE_POD` environment variable to the name of your source pod:

```
$ export SOURCE_POD=$(kubectl get pod -l app=sleep -o jsonpath={.items..metadata.name})
```

- Create a namespace for a source pod without Istio control:

```
$ kubectl create namespace without-istio
```

- Start the `sleep` sample in the `without-istio` namespace.

```
$ kubectl apply -f @samples/sleep/sleep.yaml@ -n without-istio
```

- To send requests, create the `SOURCE_POD_WITHOUT_ISTIO` environment variable to store the name of the source pod:

```
$ export SOURCE_POD_WITHOUT_ISTIO="$(kubectl get pod -n wit
hout-istio -l app=sleep -o jsonpath={.items..metadata.name}
)"
```

- Verify that the Istio sidecar was not injected, that is the pod has one container:

```
$ kubectl get pod "$SOURCE_POD_WITHOUT_ISTIO" -n without-is
tio
NAME                        READY   STATUS    RESTARTS   AGE
sleep-66c8d79ff5-8tqrl      1/1     Running   0          32s
```

Kubernetes ExternalName service to access an external service

1. Create a Kubernetes `ExternalName` service for `httpbin.org` in the default namespace:

```
$ kubectl apply -f - <<EOF
kind: Service
apiVersion: v1
metadata:
  name: my-httpbin
spec:
  type: ExternalName
  externalName: httpbin.org
  ports:
  - name: http
    protocol: TCP
    port: 80
EOF
```

2. Observe your service. Note that it does not have a cluster IP.

```
$ kubectl get svc my-httpbin
NAME         TYPE           CLUSTER-IP   EXTERNAL-IP   PORT
(S)   AGE
my-httpbin   ExternalName   <none>       httpbin.org   80/T
CP    4s
```

3. Access `httpbin.org` via the Kubernetes service's hostname from the source pod without Istio sidecar. Note that the *curl* command below uses the Kubernetes DNS format for services: `<service name>.<namespace>.svc.cluster.local`.

```
$ kubectl exec "$SOURCE_POD_WITHOUT_ISTIO" -n without-istio
 -c sleep -- curl -sS my-httpbin.default.svc.cluster.local/
headers
{
  "headers": {
    "Accept": "*/*",
    "Host": "my-httpbin.default.svc.cluster.local",
    "User-Agent": "curl/7.55.0"
  }
}
```

4. In this example, unencrypted HTTP requests are sent to `httpbin.org`. For the sake of the example only, you disable the TLS mode and allow the unencrypted traffic to the external service. In the real life scenarios, we recommend to perform Egress TLS origination by Istio.

```
$ kubectl apply -f - <<EOF
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: my-httpbin
spec:
  host: my-httpbin.default.svc.cluster.local
  trafficPolicy:
    tls:
      mode: DISABLE
EOF
```

5. Access `httpbin.org` via the Kubernetes service's hostname from the source pod with Istio sidecar. Notice the headers added by Istio sidecar, for example `X-Envoy-Decorator-Operation`. Also note that the `Host` header equals to your service's

hostname.

```
$ kubectl exec "$SOURCE_POD" -c sleep -- curl -sS my-httpbi
n.default.svc.cluster.local/headers
{
  "headers": {
    "Accept": "*/*",
    "Content-Length": "0",
    "Host": "my-httpbin.default.svc.cluster.local",
    "User-Agent": "curl/7.64.0",
    "X-B3-Sampled": "0",
    "X-B3-Spanid": "5795fab599dca0b8",
    "X-B3-Traceid": "5079ad3a4af418915795fab599dca0b8",
    "X-Envoy-Decorator-Operation": "my-httpbin.default.svc.
cluster.local:80/*",
    "X-Envoy-Peer-Metadata": "...",
    "X-Envoy-Peer-Metadata-Id": "sidecar~10.28.1.74~sleep-6
bdb595bcb-drr45.default~default.svc.cluster.local"
  }
}
```

Cleanup of Kubernetes ExternalName service

```
$ kubectl delete destinationrule my-httpbin
$ kubectl delete service my-httpbin
```

Use a Kubernetes service with endpoints to access an external service

1. Create a Kubernetes service without selector for Wikipedia:

   ```
   $ kubectl apply -f - <<EOF
   kind: Service
   apiVersion: v1
   metadata:
     name: my-wikipedia
   spec:
     ports:
     - protocol: TCP
       port: 443
       name: tls
   EOF
   ```

2. Create endpoints for your service. Pick a couple of IPs from the Wikipedia ranges list.

```
$ kubectl apply -f - <<EOF
kind: Endpoints
apiVersion: v1
metadata:
  name: my-wikipedia
subsets:
  - addresses:
      - ip: 91.198.174.192
      - ip: 198.35.26.96
    ports:
      - port: 443
        name: tls
EOF
```

3. Observe your service. Note that it has a cluster IP which you can use to access `wikipedia.org`.

```
$ kubectl get svc my-wikipedia
NAME           TYPE        CLUSTER-IP       EXTERNAL-IP   P
ORT(S)   AGE
my-wikipedia   ClusterIP   172.21.156.230   <none>        4
43/TCP   21h
```

4. Send HTTPS requests to `wikipedia.org` by your Kubernetes service's cluster IP from the source pod without Istio sidecar. Use the `--resolve` option of `curl` to access `wikipedia.org` by the cluster IP:

```
$ kubectl exec "$SOURCE_POD_WITHOUT_ISTIO" -n without-istio
 -c sleep -- curl -sS --resolve en.wikipedia.org:443:"$(kub
ectl get service my-wikipedia -o jsonpath='{.spec.clusterIP
}')" https://en.wikipedia.org/wiki/Main_Page | grep -o "<ti
tle>.*</title>"
<title>Wikipedia, the free encyclopedia</title>
```

5. In this case, the workload send HTTPS requests (open TLS connection) to the `wikipedia.org`. The traffic is already encrypted by the workload so you can safely disable Istio’s mutual TLS:

   ```
   $ kubectl apply -f - <<EOF
   apiVersion: networking.istio.io/v1alpha3
   kind: DestinationRule
   metadata:
     name: my-wikipedia
   spec:
     host: my-wikipedia.default.svc.cluster.local
     trafficPolicy:
       tls:
         mode: DISABLE
   EOF
   ```

6. Access `wikipedia.org` by your Kubernetes service's cluster IP from the source pod with Istio sidecar:

   ```
   $ kubectl exec "$SOURCE_POD" -c sleep -- curl -sS --resolve
    en.wikipedia.org:443:"$(kubectl get service my-wikipedia -
   o jsonpath='{.spec.clusterIP}')" https://en.wikipedia.org/w
   iki/Main_Page | grep -o "<title>.*</title>"
   <title>Wikipedia, the free encyclopedia</title>
   ```

7. Check that the access is indeed performed by the cluster IP. Notice the sentence `Connected to en.wikipedia.org (172.21.156.230)` in the output of `curl -v`, it mentions the IP that was printed in the output of your service as the cluster IP.

```
$ kubectl exec "$SOURCE_POD" -c sleep -- curl -sS -v --resolve en.wikipedia.org:443:"$(kubectl get service my-wikipedia -o jsonpath='{.spec.clusterIP}')" https://en.wikipedia.org/wiki/Main_Page -o /dev/null
* Added en.wikipedia.org:443:172.21.156.230 to DNS cache
* Hostname en.wikipedia.org was found in DNS cache
*   Trying 172.21.156.230...
* TCP_NODELAY set
* Connected to en.wikipedia.org (172.21.156.230) port 443 (#0)
...
```

Cleanup of Kubernetes service with endpoints

```
$ kubectl delete destinationrule my-wikipedia
$ kubectl delete endpoints my-wikipedia
$ kubectl delete service my-wikipedia
```

Cleanup

1. Shutdown the `sleep` service:

   ```
   $ kubectl delete -f @samples/sleep/sleep.yaml@
   ```

2. Shutdown the `sleep` service in the `without-istio` namespace:

```
$ kubectl delete -f @samples/sleep/sleep.yaml@ -n without-istio
```

3. Delete `without-istio` namespace:

    ```
    $ kubectl delete namespace without-istio
    ```

4. Unset the environment variables:

    ```
    $ unset SOURCE_POD SOURCE_POD_WITHOUT_ISTIO
    ```